UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/673,261	09/30/2003	Roman Talyansky	P-61115-US	2072

49443 7590 03/14/2007
PEARL COHEN ZEDEK LATZER, LLP
1500 BROADWAY 12TH FLOOR
NEW YORK, NY 10036

EXAMINER	
FENNEMA, ROBERT E	
ART UNIT	**PAPER NUMBER**
2183	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	03/14/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/673,261	TALYANSKY ET AL.	
	Examiner	Art Unit	
	Robert E. Fennema	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *04 January 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-25* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-25* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-25 have been considered. Claims 1-23 amended as per Applicant's request.

Claim Rejections - 35 USC § 112

2. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

3. Claims 8-10, 15, 17-19, 21, and 23-25 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention. Each of these claims disclose storing an index in a free register, overwriting that value with a value, and then allocating based on the index which was overwritten, and thus can not be used to allocate. One of ordinary skill in the art would not be able to make and use an invention which requires the use of a value which is purposefully deleted prior to needing and using it without undue experimentation. For the remainder of this office action, it will be assumed that the value was read into a second register such as done in Claim 5, in order to properly examine the claims.

4. Claim 5 is rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention. Claim 5 teaches that a predicate register may be freed, however, Claim 1 has explicitly stated that the instrumentation fragment has access to only one free register. When a second register is freed, there is a conflict in language, because Claim 1 states that the fragment has access to only one register, and Claim 5 states that the fragment now has access to two registers, which is in clear contradiction of each other, and one of ordinary skill in the art would not be able to make or use a method which requires the use of two registers, while only having access to one register.

5. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

6. Claims 5, 8-10, 15, 17,19, 21, and 23-25 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. The Examiner is unclear as to what the Applicant is attempting to claim as his invention, as the specification seems to indicate that the invention should only use a single free register to fulfill the allocation, however, in Claim 5, a second register (a predicate register) is used, and in the other Claims, the functionality of the claims can not be achieved using only one register, as explained above. While none of the claims have explicitly disclosed

using only a single register, rather that one register is used, if the Applicant was to make such a change, the device would appear to be inoperable, given that more than one register has to be used. Given that the specification discloses that the invention appears to use only a single register, and that the claims disclose requiring multiple registers, it is unclear what the Applicant considers to be his invention.

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

8. Claims 14 and 20 are rejected under 35 U.S.C. 102(e) as being anticipated by Connor.

9. As per Claim 14, Connor teaches: A device comprising a processor with a register stack architecture (Abstract), said device capable of allocating a spill cell using only one free register (Column 26, Lines 16-17, the OPTOP pointer is stored in a register, and the spill cell can be allocated using only it as an index).

10. As per Claim 20, Connor teaches: A system comprising:

a dynamic random access memory storage unit (Column 24, Lines 64-67); and

a processor with a register stack architecture (Abstract) capable of allocating a spill cell using only one free register (Column 26, Lines 16-17, the OPTOP pointer is stored in a register, and the spill cell can be allocated using only it as an index).

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

12. Claim 1 is rejected under 35 U.S.C. 103(a) as being unpatentable over O'Connor et al. (USPN 6,950,923, herein Connor), in view of Shande et al. ("Integrated Tool Capabilities for Performance Instrumentation and Measurement", herein Shande).

13. As per Claim 1, Connor teaches: A method comprising allocating spill cells (Column 25, Line 66 – Column 26, Line 6, it writes data to the "cells" of the stack cache, which the system sees as the stack (Column 24, Lines 53-55), thus any data arriving to it is a spill from the system) that has access to only one free register (Column 26, Lines 16-17, the OPTOP pointer is stored in a register, and the spill cell can be allocated using only it as an index) and that is run on a processor with a register stack architecture (Abstract), but fails to teach:

spill cells used by an instrumentation fragment.

While Connor teaches allocating spill cells in a register stack architecture, he does not teach that instrumentation fragments are used to cause data to be spilled. However, Shande teaches that performance analysis tools are needed in modern systems to efficiently map applications to the system (Page 1). This requires instruction fragments to be inserted into the code (Page 2), which may cause a spill to occur, as evidenced by Applicants admitted prior art, which is used as extrinsic evidence. Given the advantage of efficient mapping of applications to the system, and that Shande teaches that instrumentation is a way to do this, one of ordinary skill in the art at the time the invention was made would have been motivated to make use of instrumentation in Connor's invention in order to make it more efficient.

14. Claims 2-7 are rejected under 35 U.S.C. 103(a) as being unpatentable over Conner and Shande, further in view of Divjak ("Semaphores").

15. As per Claim 2, Connor teaches: The method of claim 1, where said allocating comprises:

designating an index for a spill array (Column 26, Lines 16-17);

incrementing said index (Column 26, Lines 19-22);

loading said incremented index in said free register (Column 26, Lines 16-17);

allocating said spill cell corresponding to said incremented index (Column 25, Line 66 – Column 26, Line 6), but fails to teach:

designating an index for a lock array;

altering a value in a cell of said lock array; and

determining whether said altered value in said cell of said lock array equals a pre-defined value.

While Conner teaches allocating spill cells, indexing the array, storing the index in a register and allocating spill cells based on the index, he does not teach a lock array which is indexed, altering the value in that array, and determining if that value equals a pre-defined value. However, Divjak teaches that semaphores in System V IPC are stored in a semaphore array, which requires an index to access. As Divjak teaches, and is well known in the art, semaphores are used to implement critical regions, or ensure that shared resources are only accessed by one process at a time in multi-process systems (Page 1). The semaphore operation sets a value in a semaphore checks to see if the value is a predetermined value, and allows access if it does. Given that memory is a shared resource, and that semaphores or some other type of locking mechanism is required for processors to execute correctly, one of ordinary skill in the art at the time the invention was made would have been motivated to use a semaphore array in the system Connor teaches, which would require an index corresponding to the spill array to use the appropriate semaphore.

16. As per Claim 3, Connor teaches: The method of claim 2, further comprising reducing said incremented index by the number of cells in said lock array if said incremented index exceeds the number of cells in said lock array (Figure 7, when an array is circular, when the bounds are exceeded, the value is reduced by the length of

the array in order to point to the bottom).

17. As per Claim 4, Divjak teaches: The method of claim 2, wherein said incrementing said index comprises executing a threadsafe instruction (Pages 1-2, a successful semaphore operation is required to use the shared resource, making using the resource threadsafe).

18. As per Claim 5, Divjak teaches: The method of claim 2, wherein said determining whether said altered value equals a pre-defined value comprises freeing a predicate register (Page 1, the value is tested against a predetermined value, and must be stored in some register to do so).

19. As per Claim 6, Divjak teaches: The method of claim 2, wherein said altering said value in said lock array comprises incrementing said value (Paragraph 1).

20. As per Claim 7, Divjak teaches: The method of claim 2, comprising reducing said altered value of said lock array if said altered value equals a maximum permitted value (Page 1, when the semaphore is released, the value is set to 0, you can't release a semaphore without first acquiring it (having the value at its maximum)).

21. Claims 8-13, 15-19, and 21-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Connor, in view of Divjak.

22. As per Claim 8, Connor teaches: A method using only one free register of a processor comprising:

storing an incremented index in the free register of a processor, such processor using a register stack architecture (Column 26, Lines 16-17, the OPTOP pointer is stored in a register, and the spill cell can be allocated using only it as an index);

allocating a cell of a second array corresponding to said index (Column 25, Line 66 – Column 26, Line 6), but fails to teach:

calculating in said free register the address of a cell of a first array corresponding to said incremented index, said cell storing a value equal to said index;

loading in said free register an incremented value from said cell of said first array;

comparing said incremented value in said free register to a pre-defined value; and

allocating if said incremented value equals said predefined value.

While Connor teaches a system which uses a stack architecture, an index to index the stack (and a stack cache), and allocating into the stack (second array) corresponding to that index, he does not teach calculating an address of a first array, loading a value from that array, and checking an incremented value from it, and allocating based on that comparison. However, Divjak teaches that semaphores in System V IPC are stored in a semaphore array, which requires an index to access. As Divjak teaches, and is well known in the art, semaphores are used to implement critical regions, or ensure that shared resources are only accessed by one process at a time in

multi-process systems (Page 1). The semaphore operation sets a value in a semaphore checks to see if the value is a predetermined value (which would require the value to be stored at least temporarily in some register), and allows access if it does. It can be seen from the semaphore data structure semid_ds that the data structure stores the index value, since the array itself must be indirectly accessed through this structure, thus the "cell" is the semaphore data structure and the cell of the array it accesses, the unit as a whole. Given that memory is a shared resource, and that semaphores or some other type of locking mechanism is required for processors to execute correctly, one of ordinary skill in the art at the time the invention was made would have been motivated to use a semaphore array in the system Connor teaches.

23. As per Claim 9, Connor teaches: The method of claim 8, comprising reducing said incremented index modulo to the number of cells in said first array (Figure 7, when an array is circular, when the bounds are exceeded, the value is reduced by the modulo length of the array in order to point to the bottom).

24. As per Claim 10, Divjak teaches: The method of claim 8, comprising reducing said incremented value if said incremented value equals a maximum permitted value (Page 1, when the semaphore is released, the value is set to 0, you can't release a semaphore without first acquiring it (having the value at its maximum)).

25. As per Claim 11, Connor teaches: A method of spill cell allocation using only one free register of a processor that uses a register stack architecture comprising:

allocating a spill cell (Column 26, Lines 16-17, the OPTOP pointer is stored in a register, and the spill cell can be allocated using only it as an index), but fails to teach:

storing an incremented value in a memory and in the free register;

comparing said incremented value in said free register to a pre-defined value;

allocating if said incremented value in said free register equals said pre-defined value; and

re-setting said incremented value in said memory.

While Connor teaches a system which uses a stack architecture and allocating into the stack, he does not teach storing an incremented value in a memory and register, comparing that value to a pre-defined value, allocating based on the comparison, and storing the compared data back into memory. However, Divjak teaches that semaphores in System V IPC are stored in a semaphore array, which requires an index to access. As Divjak teaches, and is well known in the art, semaphores are used to implement critical regions, or ensure that shared resources are only accessed by one process at a time in multi-process systems (Page 1). The semaphore operation sets a value in a semaphore checks to see if the value is a predetermined value (which would require the value to be stored at least temporarily in some register), and allows access if it does (the value must be stored back in the semaphore at some point). Given that memory is a shared resource, and that semaphores or some other type of locking mechanism is required for processors to

execute correctly, one of ordinary skill in the art at the time the invention was made would have been motivated to use a semaphore array in the system Connor teaches.

26. As per Claim 12, Divjak teaches: The method of claim 11, comprising determining if said incremented value equals a maximum permitted value (Paragraph 1, testing to see if they got the semaphore or not).

27. As per Claim 13, Divjak teaches: The method of claim 11, comprising reducing said incremented value if said incremented value equals a maximum permitted value (Page 1, when the semaphore is released, the value is set to 0, you can't release a semaphore without first acquiring it (having the value at its maximum)).

28. As per Claim 15, Connor teaches: The device of claim 14, said processor to:

store an incremented index of an array in said free register (Column 26, Lines 16-17);

allocate a spill cell of a spill array corresponding to said index in said free register (Column 25, Line 66 – Column 26, Line 6), but fails to teach:

calculate in said free register the address of a cell of an array corresponding to said incremented index, said cell storing a value equal to said index;

load in said free register an incremented value from said cell of said array;

compare said incremented value in said free register to a pre-defined value; and

allocate if said incremented value equals said pre-defined value.

While Connor teaches a system which uses a stack architecture, an index to index the stack (and a stack cache), and allocating into the stack (second array) corresponding to that index, he does not teach calculating an address of a first array, loading a value from that array, and checking an incremented value from it, and allocating based on that comparison. However, Divjak teaches that semaphores in System V IPC are stored in a semaphore array, which requires an index to access. As Divjak teaches, and is well known in the art, semaphores are used to implement critical regions, or ensure that shared resources are only accessed by one process at a time in multi-process systems (Page 1). The semaphore operation sets a value in a semaphore checks to see if the value is a predetermined value (which would require the value to be stored at least temporarily in some register), and allows access if it does. It can be seen from the semaphore data structure semid_ds that the data structure stores the index value, since the array itself must be indirectly accessed through this structure, thus the "cell" is the semaphore data structure and the cell of the array it accesses, the unit as a whole. Given that memory is a shared resource, and that semaphores or some other type of locking mechanism is required for processors to execute correctly, one of ordinary skill in the art at the time the invention was made would have been motivated to use a semaphore array in the system Connor teaches.

29. As per Claim 16, Divjak teaches: The device of claim 15, said processor further to determine if said incremented value equals a maximum permitted value (Paragraph

1, testing to see if they got the semaphore or not).

30. As per Claim 17, Divjak teaches: An article comprising a storage medium having stored thereon instructions that, when executed by a processor, result in:

storing an incremented index of an array in a free register of a processor (Column 26, Lines 16-17) using a register stack architecture (Abstract);

allocating a spill cell of a spill array corresponding to said index in said free register (Column 25, Line 66 – Column 26, Line 6), but fails to teach:

calculating in said free register the address of a cell of an array corresponding to said incremented index, said cell storing a value equal to said index;

loading in said free register an incremented value from said cell of said array; comparing said incremented value in said free register to a pre-defined value; and

allocating if said incremented value equals said predefined value.

While Connor teaches a system which uses a stack architecture, an index to index the stack (and a stack cache), and allocating into the stack (second array) corresponding to that index, he does not teach calculating an address of a first array, loading a value from that array, and checking an incremented value from it, and allocating based on that comparison. However, Divjak teaches that semaphores in System V IPC are stored in a semaphore array, which requires an index to access. As Divjak teaches, and is well known in the art, semaphores are used to implement critical regions, or ensure that shared resources are only accessed by one process at a time in multi-process systems (Page 1). The semaphore operation sets a value in a semaphore

checks to see if the value is a predetermined value (which would require the value to be stored at least temporarily in some register), and allows access if it does. It can be seen from the semaphore data structure semid_ds that the data structure stores the index value, since the array itself must be indirectly accessed through this structure, thus the "cell" is the semaphore data structure and the cell of the array it accesses, the unit as a whole. Given that memory is a shared resource, and that semaphores or some other type of locking mechanism is required for processors to execute correctly, one of ordinary skill in the art at the time the invention was made would have been motivated to use a semaphore array in the system Connor teaches.

31. As per Claim 18, Divjak teaches: The article of claim 17, wherein said instructions further result in determining if said incremented value equals a maximum permitted value (Paragraph 1, testing to see if they got the semaphore or not).

32. As per Claim 19, Divjak teaches: The article of claim 18, wherein said instructions further result in reducing said incremented value if said incremented value equals a maximum permitted value (Page 1, when the semaphore is released, the value is set to 0, you can't release a semaphore without first acquiring it (having the value at its maximum)).

33. As per Claim 21, Connor teaches: The system of claim 20, said processor to store in said free register an incremented index of an array (Column 26, Lines 16-17);

allocate said spill cell of a spill array corresponding to said index (Column 25, Line 66 – Column 26, Line 6), but fails to teach:

calculate in said free register the address of a cell of an array corresponding to said incremented index, said cell storing a value equal to said index;

load in said free register an incremented value from said cell of said array;

compare said incremented value in said free register to a pre-defined value; and

allocating if said incremented value equals said pre-defined value.

While Connor teaches a system which uses a stack architecture, an index to index the stack (and a stack cache), and allocating into the stack (second array) corresponding to that index, he does not teach calculating an address of a first array, loading a value from that array, and checking an incremented value from it, and allocating based on that comparison. However, Divjak teaches that semaphores in System V IPC are stored in a semaphore array, which requires an index to access. As Divjak teaches, and is well known in the art, semaphores are used to implement critical regions, or ensure that shared resources are only accessed by one process at a time in multi-process systems (Page 1). The semaphore operation sets a value in a semaphore checks to see if the value is a predetermined value (which would require the value to be stored at least temporarily in some register), and allows access if it does. It can be seen from the semaphore data structure semid_ds that the data structure stores the index value, since the array itself must be indirectly accessed through this structure, thus the "cell" is the semaphore data structure and the cell of the array it accesses, the unit as a whole. Given that memory is a shared resource, and that semaphores or some other

type of locking mechanism is required for processors to execute correctly, one of ordinary skill in the art at the time the invention was made would have been motivated to use a semaphore array in the system Connor teaches.

34. As per Claim 22, Divjak teaches: The system of claim 20, said processor to determine if said incremented value equals a maximum permitted value (Paragraph 1, testing to see if they got the semaphore or not).

35. As per Claim 23, Connor teaches: A processor to: store an incremented index in a free register of said processor (Column 26, Lines 16-17), such processor using a register stack architecture (Abstract);

allocate a cell of a second array corresponding to said index (Column 25, Line 66 – Column 26, Line 6), but fails to teach:

calculate in said free register the address of a cell of a first array corresponding to said incremented index, said cell storing a value equal to said index;

load in said free register an incremented value from said cell of said first array;

compare said incremented value in said free register to a pre-defined value; and

allocating if said incremented value equals said predefined value.

While Connor teaches a system which uses a stack architecture, an index to index the stack (and a stack cache), and allocating into the stack (second array) corresponding to that index, he does not teach calculating an address of a first array, loading a value from that array, and checking an incremented value from it, and

allocating based on that comparison. However, Divjak teaches that semaphores in System V IPC are stored in a semaphore array, which requires an index to access. As Divjak teaches, and is well known in the art, semaphores are used to implement critical regions, or ensure that shared resources are only accessed by one process at a time in multi-process systems (Page 1). The semaphore operation sets a value in a semaphore checks to see if the value is a predetermined value (which would require the value to be stored at least temporarily in some register), and allows access if it does. It can be seen from the semaphore data structure semid_ds that the data structure stores the index value, since the array itself must be indirectly accessed through this structure, thus the "cell" is the semaphore data structure and the cell of the array it accesses, the unit as a whole. Given that memory is a shared resource, and that semaphores or some other type of locking mechanism is required for processors to execute correctly, one of ordinary skill in the art at the time the invention was made would have been motivated to use a semaphore array in the system Connor teaches.

36. As per Claim 24, Connor teaches: The processor of claim 23, the processor to reduce said incremented index modulo to the number of cells in said first array (Figure 7, when an array is circular, when the bounds are exceeded, the value is reduced by the length of the array in order to point to the bottom).

37. As per Claim 25, Divjak teaches: The processor of claim 23, the processor to reduce said incremented value if said incremented value equals a maximum permitted

value (Page 1, when the semaphore is released, the value is set to 0, you can't release a semaphore without first acquiring it (having the value at its maximum)).

Response to Arguments

38. Examiner notes that in Applicant's remarks, Applicant appears to indicate Claim 17 as a dependant claim of Claim 14, however, it is claimed as an independent claim (and has also not been amended to include the limitation of only one free register). Applicant is asked to clarify in the next response if Claim 17 is intended to be a dependant or independent claim.

39. Regarding the enablement rejection of the previous action, Examiner does not believe the amendments to the claims overcome the rejection, and Applicant is required to explain how the amendments overcome the rejections, as it still appears that the claims require the allocation of a cell corresponding to an index which no longer is accessible, because once the index has been overwritten in the free register, there is no stored indication of the index. Applicant may have intended that the index being stored in the first array's cell is sufficient, however, without the free register to index into that array, that value is still unavailable to the machine.

40. Regarding the indefinite rejection, Examiner does not believe that Applicant's explanation overcomes the confusing nature of the claim. Claim 1 explicitly indicates that there is only one free register the fragment can access, and teaches that the

allocation can be done using only one free register, and Claim 5 states that a second register is used, which is clearly contradictory to Claim 1's statement that only one register is available. If Applicant intends for Claim 1 to indicate that the fragment has access to only one free register at a point in time, and that it can use more later, than Applicant needs to clarify that, because as the claims are currently written, Examiner does not believe it is possible for the invention to function as claimed with only one register, and if it is Applicants intent to claim that more registers can be used as needed, then a clarification is in order, as well as an amendment to the abstract, which indicates that only one register is used, and if more are used, could appear to be misleading.

41. Based on the amendment to Claim 14, Examiner has withdrawn the USC 102 rejection of Claim 14 in view of Applicant's admitted prior art.

42. Regarding the remaining 102 and 103 rejections that Applicant has argued (for essentially the same reasoning, that they do not teach using only one free register), Examiner asserts that O'Connor does teach this limitation, as O'Connor only needs one register in order to allocate a location in the "spill" section, OPTOP. OPTOP is a pointer, pointing to the available spill area, and in combination with the other references, only that register is required to allocate a spill area (after checking the semaphore array), therefore, a fragment with access to only one register can allocate spill cells, with that register being OPTOP. The argument that O'Connor shows multiple other free registers

being available is not applicable, as Applicant is referring to the spill area as the free registers, which are not used for the basis of allocating one of those registers for spill data, the OPTOP pointer is the single free register that allocates.

Conclusion

43. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Robert E. Fennema whose telephone number is (571) 272-2748. The examiner can normally be reached on Monday-Friday, 8:00-5:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Robert E Fennema
Examiner
Art Unit 2183

RF

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100